AD-A069 580 BATTELLE COLUMBUS LABS OHIO F/G 19/4

DETERMINATION OF THE HIT AND KILL PROBABILITIES FOR SHOOTING TH--ETC(U)

SEP 78 S ZACKS DAAG29-76-D-0100

UNCLASSIFIED NL

1 OF 1

AD A069580

END

DATE FILMED

7-79

DDC

LEVEL

1 B.S.

TECHNICAL REPORT

DETERMINATION OF THE HIT AND KILL PROBABILITIES

FOR SHOOTING THROUGH TREES

BY

S. ZACKS

15 Sep[redacted]78

33p.

Research conducted under Contract DAAG29-76-D-0100 with Battle Columbus, Subcontract D.O. No. 0856, for USA TRASANA.

407 080

ABSTRACT

The present paper provides a methodology for the computation of the hit/kill probability of a high caliber weapon shooting in a forest is given. We assume that trees are distributed at random according to a Poisson Law. Bullets which hit trees with sufficiently high velocity can penetrate the trunks and continue towards the target. A model is provided for the determination of the distribution of the exit velocity of a bullet. Recursive method is given for the computation of successive exit distributions as functions of the initial (muzzle) velocity, the distances between the trees and their characteristics. On the basis of this recursive method the kill probabilities are computed. Numerical examples are provided as well as FORTRAN programs.

KEY WORDS

Shooting in Forests, Exit Velocity, Distributions, Poisson distribution, Recursive Computations, Order Statistics.

TABLE OF CONTENTS

1. INTRODUCTION

The present technical report provides a method of evaluating the degradation effects on large caliber weapons located in a forest and shooting at a target in the forest. The degradation effect is actually measured by the decrease in the hit and kill probabilities of these weapons, compared to those when there are no obstacles in the trajectories of the bullets. The obstacles considered here are randomly located trees of varying trunk size. The initial (muzzle) velocity of the bullets is sufficiently high to allow penetration through trees. However, the penetrating bullet loses energy and its exit velocity may be considerably smaller than the velocity at which it hits the tree. The hitting velocity depends on the initial velocity and on the distance of the tree from the origin. The exit velocity is, however, a random variable which depends on the length of the bullet's path through the trunk and the resistance of the wood (type of tree). The penetrating bullet may also be deflected from its original aimed path. In Section 2 we specify the physical model under consideration and the probabilistic assumptions. The distribution of the exist velocity is derived from this model analytically in Section 3. In Section 4 we develop a recursive method for the numerical determination of the consecutive distributions of the exist velocities from n trees ($n \geq 1$), which are located at specified distances $d_1, d_2, \ldots, d_n$ from each other, on the path of the bullet. This recursive method is particularly convenient for numerical analysis. In Section 5 we derive an analytic expression for the hit/kill probability of a round. This analytic expression requires, however, numerical methods of evaluation. We provide numerical methods which combine exact computations with some Monte Carlo estimation. This Monte Carlo estimation appears only in one stage of the numerical evaluation, replacing a complicated numerical integration. The method is not, however, a pure simulation procedure. It has the property that with a very small number of (independent) runs we attain estimates with very high precision. Two alternative procedures are compared with respect to their precision and required computing time. FORTRAN programs are provided in the appendices.

2. THE PHYSICAL AND THE PROBABILISTIC MODEL

Consider a weapon located at the origin, 0, and shooting at a target which is at range R [m]. The initial (muzzle) velocity of the bullet is v_0 [m/sec]. If there are no obstacles along the trajectory of the bullet, it will hit the target with probability $P_H(v_0,R)$. Given that the bullet hits the target, the kill probability depends on the velocity of hitting the target, which is $v_0\lambda(v_0,R)$ and on other possible factors. Let $P_K(v_0,R)$ designate the combined kill probability. This function is specified in each particular case according to the specific weapon and fighting conditions. Similarly the function $\lambda(v_0,R)$ depends on the type of weapon, etc. We will consider here, for the sake of simplicity, a linear decreasing function of R, $v_0\ \lambda(v_0,R) = v_0-\beta R$. This is a good approximation when the initial velocity, v_0, is high and R is not too large a fraction of the weapons maximum range. The method developed in the present paper can be easily generalized to other types of ballistic functions.

The problem of shooting in the forest is that of randomly placed obstacles (trees) along the path (trajectory) of the bullet. We assume that the trees are randomly located according to a Poisson Law, with a given density, μ[No. of trees/m^2]. Thus, if we consider a strip around the straight line connecting the origin with the target, of length R[m] and width 1[m], the number of trees to be found on this strip is a random variable, N, having a Poisson distribution with mean μR.

In case of $N = n$, $n \geq 1$, let $D_1, D_2, \ldots, D_n$ be the distances (in [m]) from the origin to the location of the center of the first tree; from the first tree to the center of the second, etc.

$$\sum_{i=1}^{n} D_i \leq R$$

It is well known that the location points (Figure 1) $\xi_1 = D_1$, $\xi_2 = D_1 + D_2$, ..., $\xi_n = D_1 + ... + D_n$, are random variables having a joint distribution like the order statistics in a sample of n independent and identically distributed random variables from a uniform distribution on $[0,R]$ (H. A. David [1] pp. 80).

The trees are generally of varying size. We are actually concerned with the size of the trunk at a certain height, h, above the ground. For the purpose of modeling we assume that a cut along a horizontal plan yields a circle of radius T (Figure 2). This radius is generally a random variable with a specific distribution, $F_T(t)$. The methods developed in the present paper are for a fixed radius T. We discuss at the end how the numerical procedures can be extended to cover the case of varying radius. Notice that the length of the bullets path in the trunk is

$$L = 2\,(T^2 - u^2)^{1/2} \tag{2.1}$$

where u (the distance of the path from the center) is a random variable, having a uniform distribution on the interval (0,T). Accordingly, L is a random variable having, for a given T, a distribution function (c.d.f.)

$$P_L\,(x;T) = 1 - \left[1 - \frac{x^2}{4T^2}\right]^{1/2}, \quad 0 \le x \le 2T. \tag{2.2}$$

3. THE DISTRIBUTION OF THE EXIT VELOCITY

Given an initial velocity v_0 and a tree of radius T located at a distance D from the origin, the question is what is the distribution of the exit velocity, v_1, of a bullet going through that tree. We say that the exit velocity is zero if the bullet is absorbed in the tree.

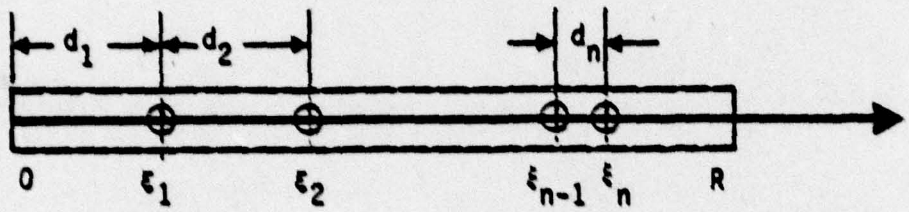

Figure 1. Random Location of Trees Along the Path of a Bullet

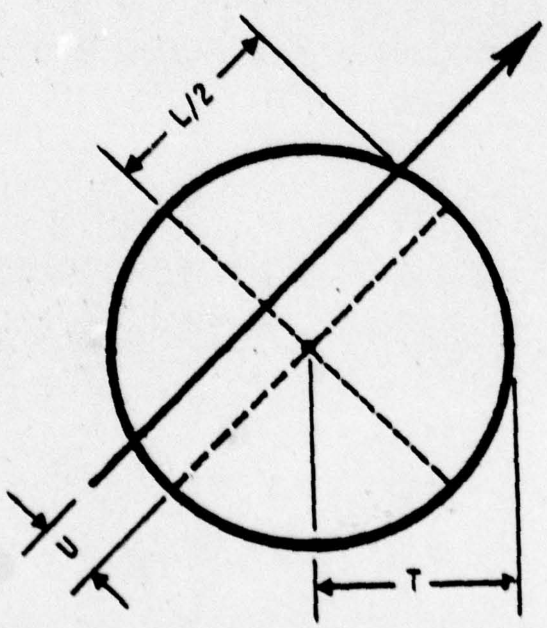

Figure 2. A Horizontal Cut of a Trunk of Radius T, With the Bullet Path

The basic physical equation is that of energy conservation, namely

$$\frac{m}{2}(\lambda(v_0,d)v_0)^2 = \frac{m}{2}v_1{}^2 + \gamma L, \tag{3.1}$$

where $d = D - T$, m is the mass of the bullet, $v_0\ \lambda(v_0,d)$ is the entrance velocity, L the length of the bullet's path within the tree and γ a proper constant, which depends on the tree's resistance (in units of $[g][m]/[sec]^2$). The physical model (3.1) is accepted to be a good first approximation to the more complicated phenomenon of a projectile penetrating a solid mass (see Lambert [3]). From (3.1) we can write

$$v_1{}^2 = (v_0\ \lambda(v_0,d))^2 - \alpha L, \tag{3.2}$$

where α is a proper constant. Let $L^*(v_0, d) = (v_0\ \lambda(v_0, d))^2/\alpha$. If $L \geq L^*$ the bullet will be absorbed in the tree. The probability of this event is, according to (2.2)

$$q(v_0,d;\ T) = \begin{cases} 0 & , \text{ if } L^*(v_0,d) \geq 2T \\ \left[1 - \left(\frac{L^*(v_0,d)}{2T}\right)^2\right]^{1/2} & , \text{ otherwise} \end{cases} \tag{3.3}$$

Similarly, the c.d.f. of the exit velocty, V_1, is

$$H(v_1;\ v_0,\ d_1,\ T) = \begin{cases} 1 & , v_1 \geq v_0\ \lambda(v_0,\ d) \\ 1 - \left[\frac{((v_0\ \lambda(v_0,d))^2 - v_1{}^2)^2}{4\alpha^2\ T^2}\right]_+^{1/2} & , 0 \leq v_1 \leq v_0\ \lambda(v_0,d) \end{cases} \tag{3.4}$$

where $[a]_+ = \max(a,o)$. Notice that if T is small, $v_o^2\lambda^2(v_o,d) - v_1^2$ may be greater than $2\alpha T$.

In these cases the c.d.f. value is zero. Thus, let

$$v_1^*(v_o,d;T) = \left[(v_o\lambda(v_o,d))^2 - 2\alpha T\right]_+^{1/2}.$$

If $q(v_o,d;T) = 0$ then $v_1^*(v_o,d;T) \geq 0$. On the other hand, if $q(v_o,d;T) > 0$ then $v_1^*(v_o,d;T) = 0$. These relationships are illustrated in Figure 3.

4. RECURSIVE DETERMINATION OF THE SUCCESSIVE EXIT VELOCITY DISTRIBUTIONS

In the following we will adopt the simple model $v_o\lambda(v_o,d) = v_o - \beta d$. This assumption is not restrictive. The following formulae are developed for this particular function, since the data showed linearity in the region of interest. The formulae can be easily modified for other types of ballistic functions. Define, $H_1(x;v_o,d,T) = H(x;v_o,d,T)$, $0 \leq x \leq v-\beta d$.

4.1 The Case of n=2

Let d_1 and d_2 be the given distances. The distribution of the exit velocity from the second tree, V_2, can be obtained from $H_1(x;v_o,d,T)$, since the exit velocity from the first tree, if given, can be applied to compute the entrance velocity into the second tree. Accordingly, the c.d.f. of V_2, given v_o, d_1, d_2 and T is

$$H_2(v;\, v_o,d_1,d_2,T) = \int_{0-}^{v_o-\beta d_1} H_1(v;x,d_2,T)\,dH_1(x;v_od_1,T) \qquad (4.1)$$

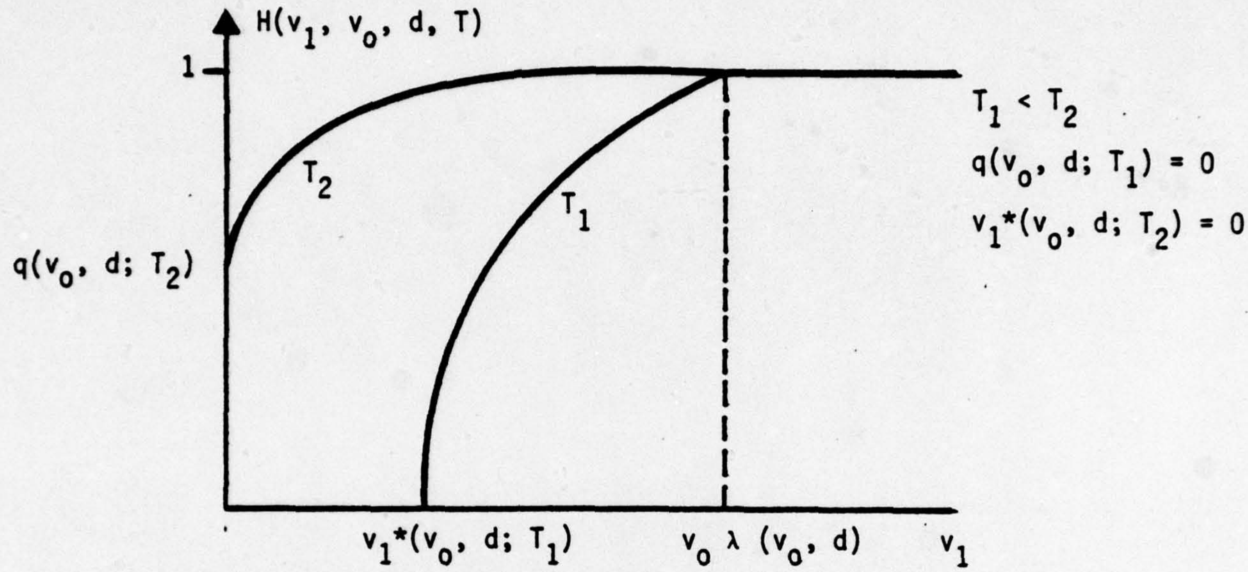

Figure 3. The C.D.F. of the Exit Velocity

Thus, we obtain after some manipulations

$$H_2(v;v_0,d_1,d_2,T) = \left[1 - \frac{[(v_0-\beta d_1-\beta d_2)^2 - v^2]^2}{4\alpha^2T^2}\right]_+^{1/2} \tag{4.2}$$

$$+ \frac{1}{2\alpha^2T^2} \int_v^{v_0-\beta d_1-\beta d_2} \left[1 - \frac{[(v_0-\beta d_1)^2 - (y+\beta d_2)^2]^2}{4\alpha^2T^2}\right]_+^{1/2} d_y \left[1 - \frac{(y^2 - v^2)^2}{4\alpha^2T^2}\right]_+^{1/2}$$

This integral should be understood as a regular integral over the range of values over which the functions within the squared brackets, [ ], are both positive. Furthermore,

$$d_y[G(y)]_+ = \begin{cases} 0 & , \text{ if } G(y) \leq 0 \\ G'(y)d_y & , \text{ if } G(y) > 0, \end{cases}$$

where $G'(y)$ is the derivative of $G(y)$. Our approach is to evaluate (4.2) numerically. We therefore leave it in its present form, without further analytical elaboration. For the purpose of approximating $H_2(v;v_0,d_1,d_2,T)$ numerically we partition the interval $(v, v_0-\beta d_1-\beta d_2)$ into M subintervals of equal size $\Delta = (v_0-\beta d_1-\beta d_2-v)/M$.

Define

$$\eta_i = v + i\Delta \qquad , \; i = 0,\, 1, \ldots M$$

$$\tilde{\eta}_i = (\eta_i + \eta_{i-1})/2 \;, \; i = 1, \ldots, M \tag{4.3}$$

We approximate then (4.2) by

$$H_2(v;v_o,d_1,d_2,T) \cong \left[1 - \frac{[(v_o-\beta d_1-\beta d_2)^2 - v^2]^2}{4\alpha^2T^2}\right]_+^{1/2} + \tag{4.4}$$

$$+ \sum_{i-1}^{m} \left[\left[1 - \frac{(\eta_{i-1}^2 - v^2)^2}{4\alpha^2T^2}\right]_+^{1/2} - \left[1 - \frac{(\eta_i^2 - v^2)^2}{4\alpha^2T^2}\right]_+^{1/2} \right] \cdot$$

$$\left[1 - \frac{[(v_o-\beta d_1)^2 - (\tilde{\eta}_i^2 + \beta d_2)^2]^2}{4\alpha^2T^2}\right]_+^{1/2}$$

As M grows (to infinity) the right hand side of (4.4) approaches that of (4.2).

In Table 1 we present the results of some computations of the c.d.f. $H_2(v;v_o,d_1,d_2,T)$ according to approximation (4.4). These computations were performed according to Program BULLET of Appendix 1, with the proper parameters and M = 400 and M = 500. We have tried the approximation also with M = 50, but for small T values (0.1 and 0.2) it has not yielded accurate results for small v values.

4.2 The General Case

After computing the values of $H_2(v;v_o,d_1,d_2,T)$, at specified values of v, one can compute $H_3(v;\ v_o,d_1,d_2,d_3,T)$ at those values of v, etc. The computation is based on the recursive formula

$$H_n\ (v_j\ v_o,\ \underset{\sim}{d}^{(n)},\ T) = \int_{0-}^{v_o-\beta\sum_{j=1}^{n-1} d_j} H_1(v;x,d_n,T)\ dH_{n-1}(x;v_o,\underset{\sim}{d}^{(n-1)},T) \quad (4.5)$$

$$= H_{n-1}(v+\beta d_n,v_o,\underset{\sim}{d}^{(n-1)},T) + \int_{v+\beta d_n}^{v_o-\beta\sum_{j=1}^{n-1} d_j} H_1(v;x,d_n,T)\ dH_{n-1}(x;v_o,\underset{\sim}{d}^{(n-1)},T)$$

For every $o \leq v \leq v_o-\beta \sum_{j-1}^{n} d_j$,

where $\underset{\sim}{d}^{(n-1)} = (d_1,\ldots,d_{n-1})_j\ \underset{\sim}{d}^{(n)} = (\underset{\sim}{d}^{(n-1)},d_n)$.

TABLE 1. The distributions $H_2(v; v_0, d_1, d_2, T)$ for v_0 = 1300 [m/sec], β = 1.8, α = 2,812,500 [m/sec^2], d_1 = 200, d_2 = 250.

	V/T	.1	.2	.3	.4	.5
	0.	0.000000	0.743070	0.966422	0.990392	0.996216
M = 400	100.	0.000000	0.743149	0.967444	0.990665	0.996321
	200.	0.000000	0.750490	0.970347	0.991447	0.996623
	300.	0.000000	0.781469	0.974685	0.992632	0.997082
	400.	0.000000	0.813191	0.979840	0.994068	0.997642
	500.	0.000000	0.864330	0.985156	0.995578	0.998234
	600.	0.000000	0.868422	0.990042	0.996997	0.998795
	700.	0.257541	0.965970	0.994063	0.998188	0.999269
	800.	0.573220	0.983473	0.996987	0.999070	0.999623
	900.	0.792740	0.993628	0.998799	0.999626	0.999848
	1000.	0.969721	0.998369	0.999685	0.999901	0.999960
	1100.	0.997259	0.999835	0.999968	0.999990	0.999996
	1200.	0.999998	1.000000	1.000000	1.000000	1.000000

PROCESSOR USAGE: 102.4 UNITS

	V/T	.1	.2	.3	.4	.5
	0.	0.000000	0.713346	0.966422	0.990392	0.996216
M = 500	100.	0.000000	0.716970	0.967445	0.990665	0.996321
	200.	0.000000	0.731379	0.970347	0.991447	0.996623
	300.	0.000000	0.765288	0.974685	0.992632	0.997082
	400.	0.000000	0.796574	0.979840	0.994068	0.997642
	500.	0.000000	0.841691	0.985156	0.995578	0.998234
	600.	0.000000	0.881748	0.990042	0.996997	0.998795
	700.	0.259115	0.965970	0.994063	0.998188	0.999269
	800.	0.556019	0.983473	0.996987	0.999070	0.999623
	900.	0.794083	0.993628	0.998799	0.999626	0.999848
	1000.	0.969721	0.998369	0.999685	0.999901	0.999960
	1100.	0.997259	0.999835	0.999968	0.999990	0.999996
	1200.	0.999998	1.000000	1.000000	1.000000	1.000000

PROCESSOR USAGE: 123.9 UNITS

Notice that $H_1(v;x,d_n,T) = 1$ for all $o \le x \le v + \beta d_n$.

The integral (4.5) is evaluated numerically, for each $n = 2,3,\ldots$ on a constant grid of v values, being $\eta_i = i\Delta$, $i = 0, 1,2,\ldots, M$ where $\Delta = 25$ [m/sec], by a formula similar to (4.4). In Tables 2 and 3 we present the numerical results of computing five exit distributions recursively. The computations were performed according to Program BUL2 given in Appendix 2. The difference between the two examples is in the value of the α coefficient.

5. DETERMINATION OF THE HIT/KILL PROBABILITIES

In Section 2 we introduced the kill probability function $P_K(v_o,R)$. If the bullet goes through n trees on its way to the target, the kill probability, given the last exit velocity $V_n = v_n$ and the vector of distances $\underset{\sim}{d}^{(n)}$, is $P_K(v_n, R-\xi_n-T)$. Accordingly, the kill probability, given $N = n$ and given $\underset{\sim}{d}^{(n)}$ is, for trees of fixed radius T and $n \geq 1$

$$\Psi_K(v_o,n,\underset{\sim}{d}^{(n)},T) = \int_0^{v_o-\beta\sum_{j=1}^{N} d_j} P_K(x,R-\xi_n-T)dH_n(x;v_o,\underset{\sim}{d}^{(n)},T). \quad (5.1)$$

Furthermore, since the conditional distribution of the points of location of the trees, $\xi_1 = D_1$, $\xi_2 = D_1+D_2,\ldots,\xi_n = D_1+\ldots,D_n$, given $N = n$, is like that of ordered statistics from a uniform distribution on (0,R), the conditional kill probability, given $N = n$ and T is for $n \geq 1$

TABLE 2. Distributions of Exit Vlocities; v_0 = 1,300 [m/sec], d_1 = 200, d_2 = 150, d_3 = 125, d_4 = 125, d_5 = 200 [m], α = 474,573.75 [m/sec^2], β = .18, T = .3 [m].

v/n	1	2	3	4	5
0.	0.000000	0.000000	0.000000	0.000000	0.000000
25.	0.000000	0.000000	0.000000	0.000000	0.000000
50.	0.000000	0.000000	0.000000	0.000000	0.000000
75.	0.000000	0.000000	0.000000	0.000000	0.000000
100.	0.000000	0.000000	0.000000	0.000000	0.000000
125.	0.000000	0.000000	0.000000	0.000000	0.000000
150.	0.000000	0.000000	0.000000	0.000000	0.000000
175.	0.000000	0.000000	0.000000	0.000000	0.000000
200.	0.000000	0.000000	0.000000	0.000000	0.000000
225.	0.000000	0.000000	0.000000	0.000000	0.000000
250.	0.000000	0.000000	0.000000	0.000000	0.000000
275.	0.000000	0.000000	0.000000	0.000000	0.000000
300.	0.000000	0.000000	0.000000	0.000000	0.000000
325.	0.000000	0.000000	0.000000	0.000000	0.000000
350.	0.000000	0.000000	0.000000	0.000000	0.000000
375.	0.000000	0.000000	0.000000	0.000000	0.000000
400.	0.000000	0.000000	0.000000	0.000000	0.000000
425.	0.000000	0.000000	0.000000	0.000000	0.005038
450.	0.000000	0.000000	0.000000	0.000000	0.009672
475.	0.000000	0.000000	0.000000	0.000000	0.030402
500.	0.000000	0.000000	0.000000	0.000000	0.057226
525.	0.000000	0.000000	0.000000	0.000000	0.082890
550.	0.000000	0.000000	0.000000	0.000000	0.130717
575.	0.000000	0.000000	0.000000	0.000000	0.185662
600.	0.000000	0.000000	0.000000	0.000000	0.244578
625.	0.000000	0.000000	0.000000	0.000000	0.318746
650.	0.000000	0.000000	0.000000	0.000000	0.415488
675.	0.000000	0.000000	0.000000	0.000000	0.510975
700.	0.000000	0.000000	0.000000	0.021790	0.599678
725.	0.000000	0.000000	0.000000	0.072218	0.675214
750.	0.000000	0.000000	0.000000	0.144083	0.757767
775.	0.000000	0.000000	0.000000	0.223350	0.825301
800.	0.000000	0.000000	0.000000	0.318064	0.879620
825.	0.000000	0.000000	0.000000	0.450056	0.920478
850.	0.000000	0.000000	0.000000	0.580120	0.949602
875.	0.000000	0.000000	0.051706	0.696111	0.969486
900.	0.000000	0.000000	0.165134	0.791503	0.983231
925.	0.000000	0.000000	0.308905	0.864775	0.991359
950.	0.000000	0.000000	0.445347	0.916221	0.995854
975.	0.000000	0.000000	0.613175	0.954470	0.998148
1000.	0.000000	0.000000	0.754902	0.977061	0.999238
1025.	0.000000	0.139957	0.858805	0.989348	0.999715
1050.	0.000000	0.394401	0.925631	0.995492	0.999905
1075.	0.000000	0.605108	0.964734	0.998291	0.999973
1100.	0.000000	0.771504	0.985291	0.999433	0.999993
1125.	0.000000	0.889233	0.994716	0.999840	0.999999
1150.	0.256790	0.955309	0.998403	0.999963	1.000000
1175.	0.647181	0.984625	0.999614	0.999994	1.000000
1200.	0.832639	0.995977	0.999936	1.000000	1.000000
1225.	0.940097	0.999407	1.000000	1.000000	1.000000
1250.	0.992331	1.000000	1.000000	1.000000	1.000000
1275.	1.000000	1.000000	1.000000	1.000000	1.000000

PROCESSOR USAGE: 20.3 UNITS
nK

TABLE 3. Distribution of Exit Velocities; v_0 = 1,300 [m/sec], d_1 = 200, d_2 = 150, d_3 = 125, d_4 = 125, d_5 = 200 [m]; α = 1,000,000 [m/sec^2], β = .18, T = .3 [m].

v/n	1	2	3	4	5
0.	0.000000	0.000000	0.248546	0.849444	0.987763
25.	0.000000	0.000000	0.249499	0.849883	0.987813
50.	0.000000	0.000000	0.252296	0.851183	0.987960
75.	0.000000	0.000000	0.256781	0.853297	0.988273
100.	0.000000	0.000000	0.262751	0.856155	0.988645
125.	0.000000	0.000000	0.269995	0.859681	0.989089
150.	0.000000	0.000000	0.278311	0.863787	0.989598
175.	0.000000	0.000000	0.298052	0.870212	0.990161
200.	0.000000	0.000000	0.313829	0.876223	0.990764
225.	0.000000	0.000000	0.329014	0.882340	0.991492
250.	0.000000	0.000000	0.344104	0.888577	0.992193
275.	0.000000	0.000000	0.371997	0.896986	0.992885
300.	0.000000	0.000000	0.393368	0.904323	0.993625
325.	0.000000	0.000000	0.413887	0.911426	0.994328
350.	0.000000	0.000000	0.448591	0.920106	0.994985
375.	0.000000	0.000000	0.473916	0.927415	0.995663
400.	0.000000	0.000000	0.509298	0.935481	0.996258
425.	0.000000	0.000000	0.538865	0.942599	0.996834
450.	0.000000	0.000000	0.575010	0.949899	0.997338
475.	0.000000	0.000000	0.606748	0.956378	0.997804
500.	0.000000	0.000000	0.644192	0.962792	0.998202
525.	0.000000	0.000000	0.676097	0.968315	0.998559
550.	0.000000	0.000000	0.714444	0.973675	0.998854
575.	0.000000	0.000000	0.744889	0.978087	0.999111
600.	0.000000	0.000000	0.781540	0.982317	0.999318
625.	0.000000	0.022092	0.813364	0.985847	0.999487
650.	0.000000	0.057398	0.840945	0.988782	0.999622
675.	0.000000	0.152739	0.868480	0.991342	0.999726
700.	0.000000	0.239789	0.891901	0.993404	0.999806
725.	0.000000	0.310658	0.912011	0.995058	0.999866
750.	0.000000	0.392621	0.930561	0.996400	0.999909
775.	0.000000	0.472353	0.945940	0.997427	0.999940
800.	0.000000	0.543983	0.958375	0.998197	0.999962
825.	0.000000	0.610063	0.969236	0.998779	0.999976
850.	0.000000	0.679238	0.977751	0.999192	0.999986
875.	0.000000	0.741204	0.984292	0.999481	0.999992
900.	0.000000	0.795375	0.989221	0.999677	0.999995
925.	0.000000	0.843923	0.992890	0.999806	0.999998
950.	0.000000	0.887755	0.995441	0.999888	0.999999
975.	0.000000	0.922429	0.997168	0.999938	0.999999
1000.	0.087551	0.947879	0.998306	0.999967	1.000000
1025.	0.410668	0.965621	0.999033	0.999984	1.000000
1050.	0.564656	0.978007	0.999480	0.999993	1.000000
1075.	0.676127	0.986585	0.999741	0.999997	1.000000
1100.	0.763202	0.992334	0.999883	0.999999	1.000000
1125.	0.832881	0.995996	0.999953	1.000000	1.000000
1150.	0.888612	0.998159	0.999984	1.000000	1.000000
1175.	0.932262	0.999302	0.999996	1.000000	1.000000
1200.	0.964843	0.999807	0.999999	1.000000	1.000000
1225.	0.986826	0.999971	1.000000	1.000000	1.000000
1250.	0.998278	1.000000	1.000000	1.000000	1.000000
1275.	1.000000	1.000000	1.000000	1.000000	1.000000

PROCESSOR USAGE: 21.0 UNITS

$F_K(v_o,n,T) =$

$$\frac{n!}{R^n}\int_0^R d\xi_n\int_0^{\xi_n} d\xi_{n-1}\cdots\int_0^{\xi_1} d\xi_1\,\Psi_K(v_o,n,\ \xi_1-T,\ \xi_2-\xi_1-2T,\ldots,\xi_n-\xi_{n-1}-2T,T) \tag{5.2}$$

where the vector $(\xi_1-T,\ \xi_2-\xi_1-T,\ldots,\ \xi_n-\xi_{n-1}-2T)$ is substituted in (5.1) for $\underset{\sim}{d}^{(n)}$. For $n = 0$ we define $F_k(v_o,\ 0,T) \equiv P_K(v_o,R)$. Finally, the total kill probability is

$$F_K(v_o,T) = e^{-R\mu}\sum_{n=0}^{\infty}\frac{(R\mu)^n}{n!}F_K(v_o,n,T). \tag{5.3}$$

We discuss here two numerical procedures for the estimation of $F_K(v_o,T)$ and their accuracy. The values of the function $P_K(x,\ R-\xi_n-T)$ are given or computed on the same grid of points $\eta_i = i\Delta$, $i = o,\ldots,25$, as that used for the numerical computation of the functions $H_n(v;v_o,\underset{\sim}{d}^{(n)},T)$. Thus, the function (5.1) is evaluated numerically according to the formula

$$\Psi_K(v_o,n,\underset{\sim}{d}^{(n)},T) \cong \tag{5.4}$$

$$\sum_{i=1}^{m} P_K(\eta_i^2,R-\Psi_n-T)[H_n(\eta_i;v_o,\underset{\sim}{d}^{(n)},T) - H_n(\eta_{i-1};v_o,\underset{\sim}{d}^{(n)},T)],\ n\geq 1.$$

It is much more complicated to evaluate the function $F_K(v_o,n,T)$ numerically. We have introduced here two methods for a Monte Carlo estimation of (5.2).

Method I

For each n, n = 1, 2,..., NP, perform the following Monte Carlo estimation independently. Simulate n uniform random variables on (0,R). Let $\xi_1, \xi_2, \ldots, \xi_n$ be the order statistics of these simulated variable. Compute the function $\Psi_K(v_0, n, \underset{\sim}{d}^{(n)}, T)$ for these values. Repeat this simulation independently NS times and average the resulting $\Psi_k(\bullet)$ values. This average $\hat{F}_K(v_0,n,T)$ is an unbiased estimator of $F_K(v_0,n,T)$. Let S_n^2 be the sample variance of the simulated $\hat{F}_k(\bullet)$ values. An estimator of the variance of $\hat{F}_K(v_0,n,T)$ is thus S_n^2/NS. Finally, the kill probability $\mathcal{F}_K(v_0,T)$ is estimated by

$$\hat{\mathcal{F}}_K(v_0,T) = e^{-\mu R} \sum_{n=0}^{NP} \frac{(\mu R)^n}{n!} \hat{F}_K(v_0,n,T),$$

$$\hat{F}_K(v_0,0,T) = F_K(v_0,0,T) = P_K(v_0,R). \tag{5.5}$$

NP is a sufficiently large integer, so that the sum of the Poisson probabilities, for n larger than NP, is sufficiently small. In the following numerical examples we specified the value of NP = INT $(\mu R + 3\sqrt{\mu R})$, where INT(x) denotes the integer smaller than or equal to x. Let pos(n; μR) denote the Poisson probability of N = n, with parameter μR. The variance of $\mathcal{F}_K(v_0,T)$ is estimated by

$$V\hat{\mathcal{F}}_K(v_0,T) = \sum_{n=0}^{NP} (\text{pos}(n;\mu R))^2 \, S_n^2/NS. \tag{5.6}$$

The square root of (5.6) is the standard-error of the estimate of the kill probability.

Method II

According to the second method we estimate $\mathcal{F}_K(v_0,T)$ in one simulation string; repeat this estimation independently NS times and take the average of the individual estimates. More specifically. Starting with $F_K(v_0,0,T) = P_K(v_0,R)$ we set

$$\hat{\mathcal{F}}_K^{(i)}(v_0,T) \longleftarrow pos(0;\mu R) * F_K(v_0,0,T).$$

We then set $n = 1$ and generate a uniform random variable from $(0,R)$. This value is set equal to d_1 and the function $\Psi_K(v_0,1,d_1,T)$ is computed.

The value of $\hat{\mathcal{F}}_K{}^{(i)}(v_0,T)$ is then changed to $\hat{\mathcal{F}}_K{}^{(i)}(v_0,T) + pos(1;R\mu)*$ $\Psi_K(v_0,1,d_1,T)$. We set then $n = 2$, generate a uniform random variable from $(0,R-d_1)$, which is set to be equal to d_2. We then compute $\Psi_K(v_0,2,d_1,d_2.T)$ and set $\hat{\mathcal{F}}_K{}^{(i)}(v_0,T) \longleftarrow \hat{\mathcal{F}}_K{}^{(i)}(v_0,T) + pos(2;\mu R)* \Psi_K(v_0,2,d_1,d_2,T)$. This simulation algorithm is continued until $n = NP$. According to this algorithm

$$\hat{\mathcal{F}}_K{}^{(i)}(v_0,T) = \sum_{N=0}^{NP} pos(n;\mu R)\, \Psi_K(v_0,n,\underset{\sim}{d}^{(n)},T). \qquad (5.7)$$

i designates the index of the simulation run, i = 1,...,NS. Finally, $F_K(v_o,T)$ is estimated by

$$\hat{\hat{F}}_K(v_o,T) = \frac{1}{NS} \sum_{i=1}^{NS} \hat{F}_K^{(i)}(v_o,T). \tag{5.8}$$

An estimate of the variance of $\hat{\hat{F}}_K(v_o,T)$ is obtained by computing the sample variance of the $\hat{F}_K^{(i)}(v_o,T)$ values (i = 1,...,NS) and dividing this sample variance by NS. The square-root of this variance is the standard error of $\hat{F}_K(v_o,T)$. In Appendix 3 we provide Program BUL6 which estimates the kill probabilities $F_K(v_o,T)$ according to Method I. Program BUL7 given in Appendix 4 is designed for the estimation of $F_K(v_o,T)$ according to Method II. In both programs we considered the special function

$$P_K(v_o,R) = \begin{cases} 1, & \text{if } v_o-\beta R \geq v_k \\ 0, & \text{otherwise.} \end{cases} \tag{5.9}$$

In Program BUL6 this special $P_K(v_o,R)$ function is programmed in the main routine. Program BUL7 is written in a more general manner. The function $P_K(v_o,R)$ is computed as a subroutine function and can be changed without altering the main program. In Table 4 we provide estimates of the kill probability computed according to Method I and Method II with the function $P_K(v_o,R)$ given by (5.9). For Method I we present the average estimates of $\Psi_K(v_o,n,\underline{d}^{(n)},T)$ and the corresponding sample standard deviations of these estimates for n = 0,1,...,NP. The estimate $F_K(v_o,T)$ and its standard error are given, too. For Method II we present the estimates $F_K^{(i)}(v_o,T)$ for i = 1,...,NS, the average $F_K(v_o,T)$ and its standard error. We see that the accuracy of Method I is somewhat higher than that of Method II. However, Method I requires more than 5 times longer computer time than Method II. It seems justified to recommend the use of Method II.

TABLE 4. Estimates of the Kill Probability by Method I and Method II; v_o = 1,300 [m/sec], μ = .005, R = 1,000 [m], α = 474,573.75 [m/sec^2], β = .18, v_k = 500 [m/sec].

METHOD I			METHOD II	
n	$\bar{\Psi}_K(v_o,n,\underline{d}^{(n)},T)$	S_n	i	$\hat{P}_K^{(i)}(v_o,T)$
0	1.000000	0.000000	1	0.646816
1	1.000000	0.000000	2	0.632648
2	1.000000	0.000000	3	0.633372
3	1.000000	0.000000	4	0.651003
4	0.999649	0.001109	5	0.613876
5	0.796708	0.092621	6	0.615645
6	0.295945	0.039027	7	0.604895
7	0.082902	0.019183	8	0.631388
8	0.015948	0.005775	9	0.664867
9	0.001935	0.000375	10	0.692548
10	0.000415	0.000254		
11	0.000035	0.000009		

Method I: $\hat{P}_K$ = .633280; S.E.$\{\hat{P}_K\}$ = .005485; Processor Usage: 1977.9 units

Method II: $\hat{\hat{P}}_K$ = .638706; S.E.$\{\hat{\hat{P}}_K\}$ = .008206; Processor Usage: 368.5 units

6. GENERALIZATION FOR RANDOM TREE RADIUS

In the present section we indicate how the previous results can be generalized to the case of random trunk radius, T. Let $F_T(T)$ be the c.d.f. of T. Given that N = n, the corresponding values $T_1,, \ldots T_n$ are independent and identically distributed. The following modifications are needed. Let $\underset{\sim}{T}^{(n)} = (T_1, \ldots, T_n)$. The exit velocity distributions are then computed as

$$H_1(v;v_o,d_1,T_1) \quad , \text{ for } n = 1 \tag{6.1}$$

$$H_2(v;v_o,d_1,d_2,T_1,T_2) = \int H_1(v;x,d_2,T_2)dH_1(x;v_o,d_1T_1) \quad , \text{ for } n = 2 \tag{6.2}$$

and

$$H_n(v;v_o,\underset{\sim}{d}^{(n)},\underset{\sim}{T}^{(n)}) = \int H_1(v;x,d_n,T_n)dH_{n-1}(x;v_o,\underset{\sim}{d}^{(n-1)},\underset{\sim}{T}^{(n-1)}), \text{ for } n > 2. \tag{6.3}$$

Given these exit velocity distributions, we determine for each $n \geq 1$, the conditional kill probabilities

$$\Psi_K(v_o,n,\underset{\sim}{d}^{(n)},\underset{\sim}{T}^{(n)}) = \int P_K(x,R-\xi_n-T_n)dH_n(x;v_o,\underset{\sim}{d}^{(n)},\underset{\sim}{T}^{(n)}). \tag{6.4}$$

The kill probabilities, given v_o, and $\{N=n\}$ are computed then as

$$\bar{F}_K(v_o,n) = \int \cdots \int F_K(v_o,n,\underset{\sim}{t}_n) \prod_{i=1}^{n} dF_T(t_i), \tag{6.5}$$

where $F_K(v_0,n,\underset{\sim}{t}_n)$ is a generalization of the function defined in (5.2) obtained by integrating $\Psi_K(v_0,n,\ \xi_1-T,\ \xi_2-\xi_1-2T,\ldots,\ \xi_n-\xi_{n-1}-2T,\ \underset{\sim}{t}_n)$ over the simplex of $\xi_1\leq\xi_2\leq\ldots\leq\xi_n$. This integral is estimated, as in the previous section by Method I. The integral (6.5) can then be evaluated numerically either by evaluating a discrete version of it, or by simulation. If one employs Method II then (6.5) is evaluated for each n in the same string of computations. Finally, in Method I the functions $F_K(v_0)$ are computed as Poisson averages of $\bar{F}_K(v_0,n)$.

References

1. Herbert A. David
Order Statistics
John Wiley and Sons, Inc., New York, 1970.

2. Edward J. Dudewicz
Introduction to Statistics and Probability
Holt, Rinehard and Winston, New York, 1976.

3. J. P. Lambert
A Residual Velocity Predictive Model for Long Rod Penetrators (U)
Memorandum Report ARRRL-MR-02828, Ballistic Research Laboratory, Aberdeen Proving Ground, Maryland, (CONFIDENTIAL), 1978.

APPENDIX I. FORTRAN PROGRAM BULLET

(Computation of $H_2(v;\ v_0,\ d_1,\ d_2,\ T)$

```
      DIMENSION H(5)
      V0=1300.
      D1=200.
      D2=250.
      A=1750000.
      B=.19
      V1=V0-B*D1
      V2=V1-B*D2
      M=500
      AM=M
      K=13
      DO 1 I=1,K
      AI=I-1
      VI=100.*AI
      DO 20 L=1,5
      AL=L
      T=.1*AL
      WI=1.-((V2*V2-VI*VI)/(2.*A*T))**2
      IF(WI) 2,3,3
    2 WI=0.
    3 H(L)=SQRT(WI)
      W1S=V1*V1-2.*A*T
      IF(W1S) 40,41,41
   40 W1S=0.
   41 V1S=SQRT(W1S)
      VS=V1S-B*D2
      IF(VI-VS) 42,43,43
   42 RI=V2-VS
      YS=VS
      GO TO 50
   43 RI=V2-VI
      YS=VI
   50 DO 4 J=1,M
      AJ=J
      YJ=YS+AJ*RI/AM
      YJJ=YJ-RI/AM
      YJM=(YJ+YJJ)/2.
      ZJ=((YJ*YJ-VI*VI)/(2.*A*T))**2
      ZJJ=((YJJ*YJJ-VI*VI)/(2.*A*T))**2
      IF(ZJ-1.) 5,4,4
    5 IF(ZJJ-1.) 6,4,4
    6 GJ=SQRT(1.-ZJ)-SQRT(1.-ZJJ)
      UJ=((V1*V1-(YJM+B*D2)**2)/(2.*A*T))**2
      IF(UJ-1.) 7,4,4
    7 TJ=SQRT(1.-UJ)
      H(L)=H(L)-TJ*GJ
    4 CONTINUE
   20 CONTINUE
      PRINT 10,VI,(H(L),L=1,5)
   10 FORMAT(5X,F6.0,5F10.6)
    1 CONTINUE
      END
```

VO ← v_0

A ← α

B ← β

T ← trunk radius

APPENDIX II. FORTRAN PROGRAM BUL2

(Recursive computation of $H_n(v;\ v_0,\ \underset{\sim}{d}^{(n)},\ T)$)

E ← Δ

A ← α

```
      DIMENSION H(5,52),D(5),U(5)
      DATA (D(I),I=1,5)/200.,150.,125.,125.,200./
      E=25.
      K=5
      V0=1300.
      A=1000000.
      T=.3
      B=.18
      M=52
      AM=M
      V1=V0-D(1)*B
      U(1)=V1
      DO 100 I=1,M
      AI=I-1
      VI=AI*E
      ZI=(V1*V1-VI*VI)/(2.*A*T)
      IF(ZI) 101,101,102
  101 H(1,I)=1.
      GO TO 100
  102 IF(ZI-1.) 103,103,104
  103 H(1,I)=SQRT(1.-ZI*ZI)
      GO TO 100
  104 H(1,I)=0.
  100 CONTINUE
      DO 200 L=2,K
      U(L)=U(L-1)-D(L)*B
      VL=U(L)
      DO 300 I=1,M
      AI=I-1
      VI=AI*E
      IF(VI-VL) 301,301,302
  301 H(L,I)=0.
      DO 400 J=1,M
      AJ=J-1
      XJ=E*(AJ+.5)
      ZJ=(XJ-D(L)*B)*(XJ-D(L)*B)-VI*VI
      ZJ=ZJ/(2.*A*T)
      IF(ZJ) 401,402,402
  401 PIJ=1.
      GO TO 405
  402 IF(ZJ-1.) 403,403,404
  403 PIJ=SQRT(1.-ZJ*ZJ)
      GO TO 405
  404 PIJ=0.
  405 IF(J-1) 406,406,407
  406 H(L,I)=H(L,I)+PIJ*H(L-1,J)
      GO TO 400
  407 H(L,I)=H(L,I)+PIJ*(H(L-1,J)-H(L-1,J-1))
  400 CONTINUE
      GO TO 300
  302 H(L,I)=1.
  300 CONTINUE
  200 CONTINUE
      DO 500 I=1,M
      II=I-1
      AI=II
      VI=AI*E
      PRINT 510,VI,(H(L,I),L=1,K)
  510 FORMAT(5X,F6.0,5F10.6)
  500 CONTINUE
      END
```

APPENDIX III. FORTRAN PROGRAM BUL6

(Estimation of $F_K(v_0, T)$ by Method I)

```
$LIB,RANDX,,,***
      DIMENSION S(52),H(52),D(100),U(100)
      R=1000.
      V0=1300.
      A=474573.75
      AL=5.
      AMU=R/AL
      T=.3
      VK=500.
      B=.18
      NR=100
      NP=11
      M=52
      AM=M
      MS=10
      AMS=MS
      E=25.
      Y=RAND(-1.)
      DO 1 I=1,NR
      Y=RAND(0.)
    1 CONTINUE
      K=0
      FK=POS(K,AL)
      VR=V0-B*R
      IF(VR-VK) 21,21,22
   21 PK=0.
      GO TO 23
   22 PK=1.
   23 TPK=PK*FK
      QPK=0.
      PRINT 24,K,PK,QPK
   24 FORMAT(5X,I4,2F10.6)
      DO 600 K=1,NP
      SPK=0.
      SSPK=0.
      DO 20 LS=1,NS
      W=0.
      Q=0.
      DO 2 J=1,K
      Y=RAND(0.)
      D(J)=Y*(R-W)
      W=W+D(J)
      Q=Q+B*D(J)
      U(J)=Q
```

Library function for generating uniform M.V.'s on (0,1)

A ← α

AL ← Rμ

This loop is just for generating the first NR = 100 uniform R.V.'s

```
    2 CONTINUE
      IF(K-1) 8,8,9
    8 V1=V0-U(1)
      VKK=VK+B*R-U(K)
      ZK=(V1*V1-VKK*VKK)/(2.*A*T)
      IF(ZK) 81,82,82
   81 PK=0.
      GO TO 85
   82 IF(ZK-1.) 83,83,84
   83 PK=1.-SQRT(1.-ZK*ZK)
      GO TO 85
   84 PK=1.
   85 SPK=SPK+PK
      SSPK=SSPK+PK*PK
      GO TO 20
    9 V1=V0-U(1)
      DO 100 I=1,M
      AI=I-1
      VI=E*AI
      ZI=(V1*V1-VI*VI)/(2.*A*T)
      IF(ZI) 101,101,102
  101 G(I)=1.
      GO TO 100
  102 IF(ZI-1.)103,103,104
  103 G(I)=SQRT(1.-ZI*ZI)
      GO TO 100
  104 G(I)=0.
  100 CONTINUE
      DO 200 L=2,K
      VL=V0-U(L)
      DO 300 I=1,M
      AI=I-1
      VI=E*AI
      IF(VI-VL) 301,301,302
  301 H(I)=0.
      DO 400 J=1,M
      AJ=J-1
      XJ=E*(AJ+.5)
      ZJ=(XJ-D(L)*B)*(XJ-D(L)*B)-VI*VI
      ZJ=ZJ/(2.*A*T)
      IF(ZJ) 401,402,402
  401 PIJ=1.
      GO TO 405
  402 IF(ZJ-1.) 403,403,404
  403 PIJ=SQRT(1.-ZJ*ZJ)
      GO TO 405
  404 PIJ=0.
  405 IF(J-1) 406,406,407
  406 H(I)=H(I)+PIJ*G(J)
      GO TO 400
  407 H(I)=H(I)+PIJ*(G(J)-G(J-1))
  400 CONTINUE
      GO TO 300
  302 H(I)=1.
  300 CONTINUE
      DO 500 I=1,M
      G(I)=H(I)
  500 CONTINUE
  200 CONTINUE
```

APPENDIX III (Continued)

```
      VKK=VK+B*R-U(K)
      IK=INT(VKK/E)+1
      PK=1.-(H(IK)+H(IK-1))/2.
      SPK=SPK+PK
      SSPK=SSPK+PK*PK
   20 CONTINUE
      APK=SPK/ANS
      SDK=(ANS*SSPK-SPK*SPK)/ANS
      SPK=SQRT(SDK/(ANS-1.))
      PRINT 34,K,APK,SPK
      FK=POS(K,AL)-POS(K-1,AL)
      TPK=TPK+FK*APK
      QPK=QPK+FK*FK*SPK*SPK/ANS
  600 CONTINUE
      SD=SQRT(QPK)
      PRINT 61,TPK,SD
   61 FORMAT(//,5X,7HES. PK=,F10.6,7HES. SD=,F10.6)
      END
```

Subroutine function for computing the Poisson c.d.f.

```
      FUNCTION POS(J,AL)
      I=J
      B=AL
      IF(B.GE.10.) GO TO 8
      IF(I) 1,2,3
    1 POS=0.
      GO TO 10
    2 POS=EXP(-B)
      GO TO 10
    3 POS=EXP(-B)
      F=POS
      DO 4 K=1,I
      AK=K
      F=F*B/AK
      POS=POS+F
    4 CONTINUE
      GO TO 10
    8 AI=I+.5
      ZI=(AI-B)/SQRT(B)
      POS=CNDX(ZI)
   10 RETURN
      END
```

For large AL it needs the normal c.d.f.

Subroutine function for computing the standard normal c.d.f.

```
      FUNCTION CNDX(X)
      Y=X
      ISWTCH=0
      IF(Y) 1,2,2
    1 Y=ABS(Y)
      ISWTCH=1
    2 P=.2316419
      B1=.31938153
      B2=-.35656378
      B3=1.7814779
      B4=-1.8212559
      B5=1.3302744
      T=1./(1.+P*Y)
      R=.3989423*EXP(-Y*Y/2.)
      QNDX=1.-R*(B1*T+B2*T*T+B3*T*T*T+B4*(T**4)+B5*(T**5))
      IF(ISWTCH) 3,4,3
    3 QNDX=1.-QNDX
    4 CNDX=QNDX
      RETURN
      END
```

APPENDIX IV. FORTRAN PROGRAM BUL7

(Computing $F_K(v_0, T)$ according to Method II)

```
$LIB,RANDX,,,***
      DIMENSION G(52),H(52),D(100),U(100)
      R=1000.
      BL=.083
      V0=1300.
      A=474573.75
      AL=BL*R
      T=.15
      VK=500.
      B=.18
      NR=100
      NP=INT(AL+3.*SQRT(AL))
      M=52
      AM=M
      NS=10
      ANS=NS
      E=25.
      Y=RAND(-1.)
      DO 1 I=1,NR
      Y=RAND(0.)
    1 CONTINUE
      K=0
      PK=FK(V0,R)
      TPK=PK*POS(K,AL)
      WPK=TPK
      QPK=0.
      SPK=0.
      DO 20 LS=1,NS
      TPK=WPK
      W=0.
      Q=0.
      K=1
      Y=RAND(0.)
      D(1)=Y*(R-W)
      W=W+D(1)
      Q=Q+B*D(1)
      U(1)=Q
      V1=V0-U(1)
      PK=0.
      DO 100 I=1,M
      AI=I-1
      VI=E*AI
      ZI=(V1*V1-VI*VI)/(2.*A*T)
      IF(ZI) 101,101,102
  101 G(I)=1.
      GO TO 105
  102 IF(ZI-1.)103,103,104
  103 G(I)=SQRT(1.-ZI*ZI)
      GO TO 105
  104 G(I)=0.
  105 VIT=VI+E/2.
      IF(I.GT.1) GO TO 106
      PK=PK+FK(VIT,R-W)*G(1)
      GO TO 100
```

Library function for generating uniform R.V.'s on (0,1).

BL ← μ

AL ← μR

APPENDIX IV (Continued)

```
106 PK=PK+FK(VIT,R-W)*(G(I)-G(I-1))
100 CONTINUE
    TPK=TPK+PK*(POS(K,AL)-POS(K-1,AL))
    DO 200 L=2,NP
    Y=RAND(0.)
    D(L)=Y*(R-W)
    W=W+D(L)
    Q=Q+B*D(L)
    U(L)=Q
    VL=V0-U(L)
    PK=0.
    DO 300 I=1,M
    AI=I-1
    VI=E*AI
    VIT=VI+E/2.
    IF(VI-VL) 301,301,302
301 H(I)=0.
    DO 400 J=1,M
    AJ=J-1
    XJ=E*(AJ+.5)
    ZJ=(XJ-D(L)*B)*(XJ-D(L)*B)-VI*VI
    ZJ=ZJ/(2.*A*T)
    IF(ZJ) 401,402,402
401 PIJ=1.
    GO TO 405
402 IF(ZJ-1.) 403,403,404
403 PIJ=SQRT(1.-ZJ*ZJ)
    GO TO 405
404 PIJ=0.
405 IF(J-1) 406,406,407
406 H(I)=H(I)+PIJ*G(J)
    GO TO 400
407 H(I)=H(I)+PIJ*(G(J)-G(J-1))
400 CONTINUE
    GO TO 303
302 H(I)=1.
303 RK=FK(VIT,R-W)
306 IF(I.GT.1) GO TO 304
    PK=PK+RK*H(1)
    GO TO 300
304 PK=PK+RK*(H(I)-H(I-1))
300 CONTINUE
    DO 500 I=1,M
    G(I)=H(I)
500 CONTINUE
    TPK=TPK+PK*(POS(L,AL)-POS(L-1,AL))
200 CONTINUE
    SPK=SPK+TPK
    QPK=QPK+TPK*TPK
    PRINT 24,LS,TPK
 24 FORMAT(5X,I4,F10.6)
 20 CONTINUE
    APK=SPK/ANS
    VPK=(ANS*QPK-SPK*SPK)/ANS
    SDK=SQRT(VPK/(ANS-1.)*ANS)
    PRINT 61,APK,SDK
 61 FORMAT(5X,F10.6,5X,F10.6)
    END
```

APPENDIX IV (Continued)

Subroutine function for computing the Poisson c.d.f.

```
      FUNCTION POS(J,AL)
      I=J
      B=AL
      IF(B.GE.10.) GO TO 8
      IF(I) 1,2,3
    1 POS=0.
      GO TO 10
    2 POS=EXP(-B)
      GO TO 10
    3 POS=EXP(-B)
      F=POS
      DO 4 K=1,I
      AK=K
      F=F*B/AK
      POS=POS+F
    4 CONTINUE
      GO TO 10
    8 AI=I+.5
      ZI=(AI-B)/SQRT(B)
      POS=CNDX(ZI)
   10 RETURN
      END
```

Subroutine function for computing the standard normal c.d.f.

```
      FUNCTION CNDX(X)
      Y=X
      ISWTCH=0
      IF(Y) 1,2,2
    1 Y=ABS(Y)
      ISWTCH=1
    2 P=.2316419
      B1=.31938153
      B2=-.35656378
      B3=1.7814779
      B4=-1.3212559
      B5=1.3302744
      T=1./(1.+P*Y)
      R=.3989423*EXP(-Y*Y/2.)
      QNDX=1.-R*(B1*T+B2*T*T+B3*T*T*T+B4*(T**4)+B5*(T**5))
      IF(ISWTCH) 3,4,3
    3 QNDX=1.-QNDX
    4 CNDX=QNDX
      RETURN
      END
```

Subroutine function for computing $P_K(v_0, R)$

```
      FUNCTION FK(V,R)
      W=V
      P=R
      WK=500.
      B=.18
      WS=W-P*B
      IF(WS-WK) 1,1,2
    1 FK=0.
      GO TO 10
    2 FK=1.
   10 RETURN
      END
```